|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **文档编号** | 1-4 | | **当前版本** | **1.0** |
| **当前状态** | 草稿 |  | **发布日期** | 2017-08-25 |
| 发布 | √ |

< HeatPoint可视化系统>

详细设计说明书

**拟制人：** 张仁童

**审核人：**

**批准人：**

目录

[1引言 2](#_Toc491455927)

[1.1编写目的 2](#_Toc491455928)

[1.2 项目背景与意义 2](#_Toc491455929)

[1.3背景 3](#_Toc491455930)

[1.4主要技术简介 3](#_Toc491455931)

[1.4.1爬虫策略 4](#_Toc491455932)

[1.4.2 分布式计算 6](#_Toc491455933)

[1.4.3 参考资料 7](#_Toc491455934)

[2 系统整体设计 8](#_Toc491455935)

[3 分布式爬虫设计说明 9](#_Toc491455936)

[3.1 程序描述 10](#_Toc491455937)

[3.2 功能 10](#_Toc491455938)

[3.3输入项 10](#_Toc491455939)

[3.4流程逻辑 11](#_Toc491455940)

[4 可视化设计说明 12](#_Toc491455941)

[4.1程序描述 13](#_Toc491455942)

# 1引言

## 1.1编写目的

编写详细设计说明书的目的是为了使开发人员在完成概要设计说明书的基础上，完成概要设计的各项功能规定；为程序员写出实际的程序代码提供依据。它是软件设计阶段所有任务和所有相关人员所需的参考资料。预期的读者为系统的开发者、使用者和指导老师。

## 1.2 项目背景与意义

随着互联网大数据的发展，各种大数据的分析对各行业都产生了不同程度的影响。网站数据、社交媒体数据等是互联网大数据的重要组成部分。对于民航业领域，社会事件的发生，会很大程度影响旅客的出行需求变化，从而影响航空公司飞机运力投放、航班编排、票价策略调整等，对互联网事件的准确抓取和分析能够帮助航空公司更好的服务市场、服务旅客、提升收益。

本题要求参赛选手从互联网上抓取事件，并进行分词及语义等处理，提取事件的属性建立事件模型，以及可视化展现。

对于航空公司来说，能否预先判断旅客的出行需求，可用于航空公司及行业管理部门决定行业运力投放的合理性，从而提高旅客服务水平、提高航空公司航班收益水平。目前，仅靠航空公司拥有的数据是远远不够的，从互联网中获取大量的影响民航领域的信息能够帮助航空公司制定销售策略。而互联网事件数据，是其中非常重要的一种数据来源。

通过初步分析，对民航业可能有影响的事件主要可分为如下几类：政治会议（例G20）、展会、体育赛事、演唱会、突发异常天气等。对不同类型的事件分别分析，并建立一个相对统一的模型描述事件，用来分析其对旅客出行的影响。

但是互联网上的数据复杂性非常高，一是事件数据来源多；二是各数据源对事件的描述非常复杂，来自不同数据源的同一事件，可能差别较大；三是事件数据一般为非结构化数据，其处理和分析有一定的难度。

传统做法需要投入大量人工进行标注。因此要求参赛者具有抓取数据的技术和一定的自然语言处理能力，实现自动化。

## 1.3背景

基于互联网大数据的事件智能抓取和画像是中国软件杯提出的命题，我们根据题目要求开发的一款能够智能抓取互联网上所有包含地理位置信息的内容，并分类可视化于地图上的产品，主要功能是实现互联网信息采集、事件的去重、事件画像建模、可视化展现。项目周期为13周，项目背景规划如下表所示。

表1.1 项目背景规划表

|  |  |  |  |
| --- | --- | --- | --- |
| 项目名称 | 任务提出者 | 用户 | 项目承办单位 |
| QR二维码编解码系统 | 中国软件杯主办方 | 个人企业用户 | 南京信息职业技术学院 |

## 1.4主要技术简介

网络爬虫（英语：web crawler），也叫网络蜘蛛（spider），是一种用来自动浏览[万维网](https://zh.wikipedia.org/wiki/%E4%B8%87%E7%BB%B4%E7%BD%91)的[网络机器人](https://zh.wikipedia.org/w/index.php?title=%E7%BD%91%E7%BB%9C%E6%9C%BA%E5%99%A8%E4%BA%BA&action=edit&redlink=1)。其目的一般为编纂[网络索引](https://zh.wikipedia.org/w/index.php?title=%E7%BD%91%E7%BB%9C%E7%B4%A2%E5%BC%95&action=edit&redlink=1)。

[网络](https://zh.wikipedia.org/wiki/%E7%BD%91%E7%BB%9C%E6%90%9C%E7%B4%A2%E5%BC%95%E6%93%8E)[搜索引擎](https://zh.wikipedia.org/wiki/%E6%90%9C%E7%B4%A2%E5%BC%95%E6%93%8E)等站点通过爬虫软件更新自身的[网站内容](https://zh.wikipedia.org/w/index.php?title=%E7%B6%B2%E7%AB%99%E5%85%A7%E5%AE%B9&action=edit&redlink=1)或其对其他网站的索引。网络爬虫可以将自己所访问的页面保存下来，以便搜索引擎事后生成[索引](https://zh.wikipedia.org/w/index.php?title=%E7%B4%A2%E5%BC%95_(%E6%90%9C%E7%B4%A2%E5%BC%95%E6%93%8E)&action=edit&redlink=1)供用户搜索。

爬虫访问网站的过程会消耗目标系统资源。不少网络系统并不默许爬虫工作。因此在访问大量页面时，爬虫需要考虑到规划、负载，还需要讲“礼貌”。 不愿意被爬虫访问、被爬虫主人知晓的公开站点可以使用[robots.txt](https://zh.wikipedia.org/wiki/Robots.txt)文件之类的方法避免访问。这个文件可以要求[机器人](https://zh.wikipedia.org/w/index.php?title=%E8%BD%AF%E4%BB%B6%E5%8A%A9%E7%90%86&action=edit&redlink=1)只对[网站](https://zh.wikipedia.org/wiki/%E7%BD%91%E7%AB%99)的一部分进行索引，或完全不作处理。

互联网上的页面极多，即使是最大的爬虫系统也无法做出完整的索引。因此在公元2000年之前的万维网出现初期，搜索引擎经常找不到多少相关结果。现在的搜索引擎在这方面已经进步很多，能够即刻给出高质量结果。

爬虫还可以验证[超链接](https://zh.wikipedia.org/wiki/%E8%B6%85%E9%80%A3%E7%B5%90)和[HTML](https://zh.wikipedia.org/wiki/HTML)代码，用于[网络抓取](https://zh.wikipedia.org/w/index.php?title=%E7%BD%91%E7%BB%9C%E6%8A%93%E5%8F%96&action=edit&redlink=1)（参见[数据驱动编程](https://zh.wikipedia.org/w/index.php?title=%E6%95%B0%E6%8D%AE%E9%A9%B1%E5%8A%A8%E7%BC%96%E7%A8%8B&action=edit&redlink=1)）。

网络爬虫始于一张被称作种子的统一资源地址（URL）列表。当网络爬虫访问这些统一资源定位器时，它们会甄别出页面上所有的超链接，并将它们写入一张“待访列表”，即所谓[爬行疆域](https://zh.wikipedia.org/w/index.php?title=%E7%88%AC%E8%A1%8C%E7%96%86%E5%9F%9F&action=edit&redlink=1)（crawl frontier）。此疆域上的诸URL将被按照一套策略循环访问。如果爬虫在他执行的过程中复制归档和保存网站上的信息，这些档案通常储存，使他们可以被查看。阅读和浏览他们的网站上实时更新的信息，并保存为网站的“快照”。大容量的体积意味着网络爬虫只能在给定时间内下载有限数量的网页，所以要优先考虑其下载。高变化率意味着网页可能已经被更新或者删除。一些被服务器端软件生成的URL（统一资源定位符）也使得网络爬虫很难避免检索到重复内容。

互联网资源卷帙浩繁，这意味着网络爬虫在一定时间内只能下载有限数量的网页，因此它需要优化它的下载方式。互联网资源瞬息万变，这也意味着网络爬虫下载的网页在使用前就已经被修改甚至是删除了。这是网络爬虫设计者们所面临的两个基本问题。

再者，服务器端软件所生成的统一资源地址数量庞大，以至于网络爬虫难以避免的采集到重复内容。根据超文本协议“显示请求”（HTTP GET）的参数的无尽组合所返回的页面中，只有很少一部分确实传回唯一的内容。例如：一个照片陈列室网站，可能通过几个参数，让用户选择相关照片：其一是通过四种方法对照片排序，其二是关于照片分辨率的的三种选择，其三是两种文件格式，另加一个用户可否提供内容的选择，这样对于同样的结果极可能会有48种不同的统一资源地址与其关联。这种数学组合给网络爬虫制造了麻烦，因为它们必须越过这些无关脚本变化的组合，寻找到不重复的内容。

### 1.4.1爬虫策略

爬虫的实现由以下策略组成：

* 指定页面下载的选择策略
* 检测页面是否改变的重新访问策略
* 定义如何避免网站过度访问的约定性策略
* 如何部署分布式网络爬虫的并行策略

爬虫可能只想搜索HTML页面而避免其他MIME 类型。为了只请求HTML资源，爬虫在抓取整个以GET方式请求的资源之前，通过创建HTTP的HEAD请求来决定网络资源的MIME类型。为了避免发出过多的请求，爬虫会检查URL和只请求那些以某些字符（如.html, .htm, .asp, .aspx, .php, .jsp, .jspx 或 / ）作为后缀的URL。这个策略可是跳过很多HTML网络资源。

有些爬虫还能避免请求一些带有“?”的资源（动态生成）。为了避免掉入从网站下载无限量的URL的爬虫陷阱。不过假若网站重写URL以简化URL的目的，这个策略就变得不可靠了。

##### URL规范化

爬虫通常使用某些URL规范化的方式以避免资源的重复爬取。URL规范化，指的是以某种一致的方式修改和标准化URL的过程。这个过程有各种各样的处理规则，包括统一转换为小写、移除“.”和“..”片段，以及在非空路径里插入斜杆。

##### 路径上移爬取

有些爬虫希望从指定的网站中尽可能地爬取资源。而路径上移爬虫就是为了能爬取每个URL里提示出的每个路径。例如，给定一个Http的种子URL: http://llama.org/hamster/monkey/page.html ，要爬取 /hamster/monkey/ ， /hamster/ 和 / 。Cothey发现路径能非常有效地爬取独立的资源，或以某种规律无法在站内链接爬取到的资源。

##### 主题爬取

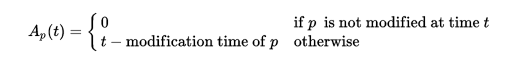
对于爬虫来说，一个页面的重要性也可以说是，给定查询条件一个页面相似性能起到的作用。网络爬虫要下载相似的网页被称为主题爬虫或局部爬虫。这个主题爬虫或局部爬虫的概念第一次被[Filippo Menczer](https://zh.wikipedia.org/w/index.php?title=Filippo_Menczer&action=edit&redlink=1) 和 Soumen Chakrabarti 等人提出的。

#### 重新访问策略

网站的属性之一就是经常动态变化，而爬取网站的一小部分往往需要花费几个星期或者几个月。等到网站爬虫完成它的爬取，很多事件也已经发生了，包括增加、更新和删除。 在搜索引擎的角度，因为没有检测这些变化，会导致存储了过期资源的代价。最常用的估价函数是新鲜度和过时性。 新鲜度：这是一个衡量抓取内容是不是准确的二元值。在时间t内，仓库中页面p的新鲜度是这样定义的：



过时性:这是一个衡量本地已抓取的内容过时程度的指标。在时间t时，仓库中页面p的时效性的定义如下：



{\displaystyle A\_{p}(t)={\begin{cases}0&{\rm {if}}~p~{\rm {~is~not~modified~at~time}}~t\\t-{\rm {modification~time~of}}~p&{\rm {otherwise}}\end{cases}}}

#### 平衡礼貌策略

爬虫相比于人，可以有更快的检索速度和更深的层次，所以，他们可能使一个站点瘫痪。不需要说一个单独的爬虫一秒钟要执行多条请求，下载大的文件。一个服务器也会很难响应多线程爬虫的请求。 就像Koster所注意的那样，爬虫的使用对很多任务作都是很有用的，但是对一般的社区，也需要付出代价。使用爬虫的代价包括：

* 网络资源：在很长一段时间，爬虫使用相当的带宽高度并行地工作。
* 服务器超载：尤其是对给定服务器的访问过高时。
* 质量糟糕的爬虫，可能导致服务器或者路由器瘫痪，或者会尝试下载自己无法处理的页面。
* 个人爬虫，如果过多的人使用，可能导致网络或者服务器阻塞。

对这些问题的局部解决方法是漫游器排除协议（Robots exclusion protocol），也被称为robots.txt议定书[[10]](https://zh.wikipedia.org/wiki/%E7%B6%B2%E8%B7%AF%E7%88%AC%E8%9F%B2#cite_note-10)，这份协议是让管理员指明网络服务器的不应该爬取的约定。这个标准没有包括重新访问一台服务器的间隔的建议，虽然设置访问间隔是避免服务器超载的最有效办法。最近的商业搜索引擎，如Google，Ask Jeeves，MSN和Yahoo可以在robots.txt中使用一个额外的 “Crawl-delay”参数来指明请求之间的延迟。

#### 并行策略

一个并行爬虫是并行运行多个进程的爬虫。它的目标是最大化下载的速度，同时尽量减少并行的开销和下载重复的页面。为了避免下载一个页面两次，爬虫系统需要策略来处理爬虫运行时新发现的URL，因为同一个URL地址，可能被不同的爬虫进程抓到。

### 1.4.2 分布式计算

在计算机科学中，分布式计算（英语：Distributed computing），又译为分散式運算。这个研究领域，主要研究分布式系统（Distributed system）如何进行计算。分布式系统是一组电脑（computer），通过网络相互链接传递消息与通信后并协调它们的行为而形成的系统。组件之间彼此进行交互以实现一个共同的目标。把需要进行大量计算的工程数据分区成小块，由多台计算机分别计算，再上传运算结果后，将结果统一合并得出数据结论的科学。分布式系统的例子来自有所不同的面向服务的架构，大型多人在线游戏，对等网络应用。

### 1.4.3 参考资料

1. Spetka, Scott. [The TkWWW Robot: Beyond Browsing](https://web.archive.org/web/20040903174942/archive.ncsa.uiuc.edu/SDG/IT94/Proceedings/Agents/spetka/spetka.html). [NCSA](https://zh.wikipedia.org/wiki/%E5%9B%BD%E5%AE%B6%E8%B6%85%E7%BA%A7%E7%94%B5%E8%84%91%E5%BA%94%E7%94%A8%E4%B8%AD%E5%BF%83). [21 November 2010]. （[原始内容](http://archive.ncsa.uiuc.edu/SDG/IT94/Proceedings/Agents/spetka/spetka.html)存档于3 September 2004）.
2. Kobayashi, M. & Takeda, K. [Information retrieval on the web](http://doi.acm.org/10.1145/358923.358934). ACM Computing Surveys (ACM Press). 2000, **32** (2): 144–173. [doi:10.1145/358923.358934](https://dx.doi.org/10.1145%2F358923.358934).
3. 参见[FOAF项目wiki对“scutter”的描述](http://wiki.foaf-project.org/w/Scutter)
4. Castillo, Carlos. [Effective Web Crawling](http://chato.cl/research/crawling_thesis) (Ph.D. thesis). University of Chile. 2004 [2010-08-03].
5. Cothey, Viv. Web-crawling reliability. Journal of the American Society for Information Science and Technology. 2004, **55** (14): 1228–1238.[doi:10.1002/asi.20078](https://dx.doi.org/10.1002%2Fasi.20078).
6. Menczer, F. (1997). [ARACHNID: Adaptive Retrieval Agents Choosing Heuristic Neighborhoods for Information Discovery](http://informatics.indiana.edu/fil/Papers/ICML.ps). In D. Fisher, ed., Machine Learning: Proceedings of the 14th International Conference (ICML97). Morgan Kaufmann
7. Menczer, F. and Belew, R.K. (1998). [Adaptive Information Agents in Distributed Textual Environments](http://informatics.indiana.edu/fil/Papers/AA98.ps). In K. Sycara and M. Wooldridge (eds.) Proc. 2nd Intl. Conf. on Autonomous Agents (Agents '98). ACM Press
8. Chakrabarti, S., van den Berg, M., and Dom, B. (1999). [Focused crawling: a new approach to topic-specific web resource discovery](https://web.archive.org/web/20040317210216/http:/www.fxpal.com/people/vdberg/pubs/www8/www1999f.pdf). Computer Networks, 31(11–16):1623–1640.
9. E. G. Coffman Jr; Zhen Liu; Richard R. Weber. Optimal robot scheduling for Web search engines. Journal of Scheduling. 1998, **1** (1): 15–29.[doi:10.1002/(SICI)1099-1425(199806)1:1<15::AID-JOS3>3.0.CO;2-K](https://dx.doi.org/10.1002%2F%28SICI%291099-1425%28199806%291%3A1%3C15%3A%3AAID-JOS3%3E3.0.CO%3B2-K).
10. Koster, M. (1996). [A standard for robot exclusion](http://www.robotstxt.org/wc/exclusion.html)
11. Coulouris, George; Jean Dollimore; Tim Kindberg; Gordon Blair. Distributed Systems: Concepts and Design (5th Edition). Boston: Addison-Wesley. 2011. [ISBN 0-132-14301-1](https://zh.wikipedia.org/wiki/Special:%E7%BD%91%E7%BB%9C%E4%B9%A6%E6%BA%90/0-132-14301-1)..

# 2 系统整体设计

本系统分为前端与后端两个部分，前端主要负责数据可视化由以下四个模块构成：地理位置、事件建模、航班信息和统计视图。后端主要负责信息收集，采用分布式爬虫处理，分为主节点和若干个分布式节点协同工作采集数据。系统结构图如下：

HeatPoint可视化系统

分布式爬虫

可视化

监听程序

主节点

控制程序

事件建模

航班信息

分布式节点

地理位置

统计视图

接收程序

持久化处理

爬虫程序

图2.1 系统总体结构图

# 3 分布式爬虫设计说明

**Crawler**

包括API接口和静态默认配置文件，详情请参考API文档。

**Entity**

是爬虫节点经过爬取后返回主节点的数据包实例，也是节点进行后续处理的单位，详情请参考API文档。

**Main**

包括主节点从初始化到向个节点发送请求并接收返回的数据包及数据相关处理的所有文件。

**Node**

依次包括：新闻博客网站的自动提取进程、爬虫节点的接收请求服务器、爬取后的数据处理接口、节点接收程序、请求分析及判断网站类型启动相应爬取程序、普通单独页面的数据爬取进程、节点服务器启动程序。单独页面爬取后将反射调用数据处理接口，进行处理及数据持久化操作，如未添加自定义处理类，则使用程序内置的数据持久化模块，详情请参考API文档。

**Utils**

所有静态工具类，依次包括：数据库访问工具、字符串处理工具、页面下载及处理工具。

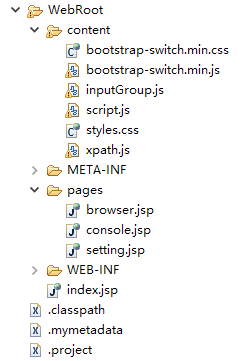
**Servlet**

爬虫控制界面的响应服务器和爬虫状态监控长连接服务器。

**Test**

文件读取、分析、输出程序和测试类。

**Content**

 爬虫控制界面的额外文件，依次包括：开关的样式、开关的动画及功能、输入框组的生成、主要脚本文件、主要样式文件、xpath自动提取工具。

**Page**

所有JSP文件，依次包括：页面预览浏览器界面、爬虫控制台界面、爬虫基本配置界面。

## 3.1 程序描述

本程序是CrawlerSYS，一个轻量级高性能智能可靠分布式爬虫系统，为本系统的重要子程序。

## 3.2 功能

数据库

分布式节点

输入爬虫配置信息

分布式节点

网络信息

分布式节点

启动爬虫主节点

## 3.3输入项

**public** **static** **final** String *dbip* = "119.29.223.149/heatpoint";

数据库网络位置

**public** **static** **final** String *dbArgs* = "useUnicode=true&characterEncoding=UTF-8";

数据库配置

**public** **static** **final** String *user* = "root";

数据库用户名

**public** **static** **final** String *password* = "root";

数据库密码

**public** **static** **final** **int** *urlLimit* = 30;

单次传输的网页量

**public** **static** **final** **int** *thread* = 10;

启用的线程数

**public** **static** **final** **int** *allLimit* = 5000;

总页面量限制

**public** **static** **final** **int** *sleepTime* = 0;

爬取后暂停时间

**public** **static** **final** **int** *serverPort* = 6545;

服务端口

**public** **static** **final** **int** *socketPort* = 6546;

通信端口

**public** **static** **final** String[] *node* = {"127.0.0.1:6545"};

网络节点位置

## 3.4流程逻辑

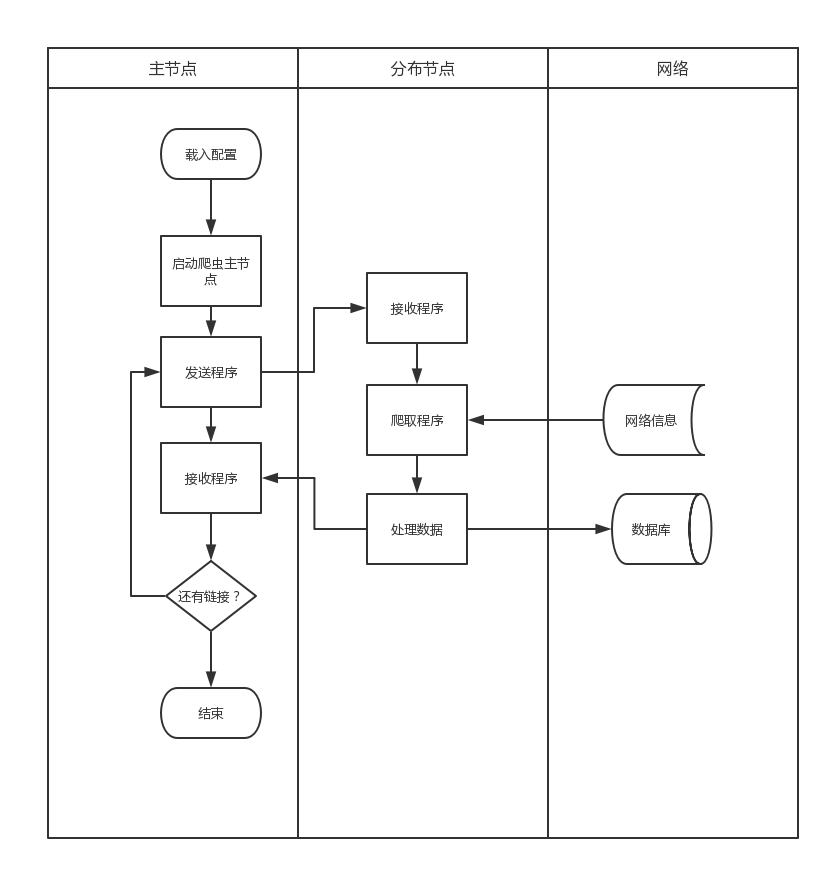
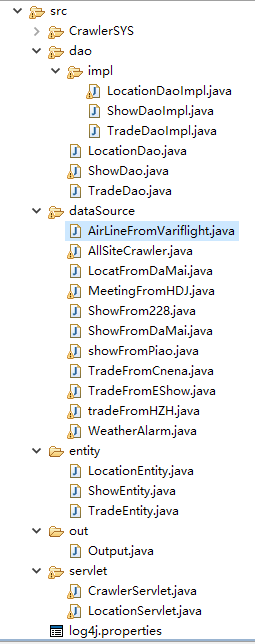


图3.4流程逻辑

4 可视化设计说明

**Dao**

数据持久化层，用于与数据库通信，分为接口和实例。依次包括：地理位置信息、其他活动信息、展会信息。

**DataSource**

自定义爬虫，从不同数据源中使用不同方法爬取不同结构的数据，具体请参见《数据库结构》。依次包括：飞常准上的航班信息、全部站点爬虫启动器、大麦网上的场馆信息、活动家上的会议信息、永乐票务上的活动信息、大麦网上的活动信息、中国会展门户上的展会信息、E展会上的展会信息、中国天气上的天气预警信息。

**Entity**

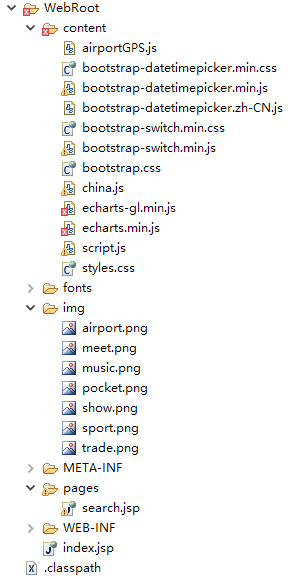
各个类间、与数据库间通信使用的实体类。

Out

从数据库中未过期数据自动化提取事件属性，并生成符合要求格式的相应文件。

Servlet

控制爬虫功能及初始化爬虫功能的服务器和查询符合所有条件的地理位置信息服务器。



Content

可视化界面的额外文件，依次包括：所有机场的地理位置信息、选择日期的样式、选择日期的动画和功能、选择日期的中文支持、开关的样式、开关的动画及功能、自定义的bootstrap样式、小型中国地图的数据、3D图表的功能、普通图表的功能、主要脚本文件、主要样式文件。

Img

所有图标文件，依次包括：机场、会议、音乐、娱乐、体育赛事、展会。

Pages

顶部搜索栏JSP页面。

# 

## 4.1程序描述

把数据库中的数据可视化到网页上，包括地图和统计视图。